1. The default constructor is simply a constructor with no arguments passed to it. It does not need to be explicitly defined so running this code will create a new instance of the Name() class with 0 parameters passed to it. If Name() has a string value, it will be automatically assigned null.
2. Questions from Test class:
   (a) The lines of code that will cause a compile-time error are the last two System.out.print lines. Static variables and methods belong to their class rather than an instantiated object and typically there is only one copy of a static variable/method for the whole class. Static methods can ONLY access static variables so when the static method "showAllTwo" tries to access y and z (which aren't static variables) it runs into scoping problems. (I'm not 100 percent sure if the problem is exactly scoping, I'm 99 percent sure. If that is incorrect please let me know).
   (b) There are two major problems here with the final variables and the static context. First off, final variables cannot be changed, so any lines of code that tries to change these variables will run into problems like on lines 2,4,8, and 10. Secondly, after line 7, the tester object is no longer calling the variables, instead the Class Test is, so variables that aren't static cannot be referenced since the Class Test is a static context (lines 10,11,12).
3. Zip file submission
4. Questions from class A and B (B extends A):
   (a) Neither objects can access *theData*
   (b) Both objects can invoke method w.
   (c) Both objects can invoke method y.
   (d) Only bInstance can invoke method r.
   (e) Neither objects can invoke z.
   (f) The method y can be invoked by all methods
   (g) The method r can only be invoked by Class B methods, so only x,r, and s

(h) The method z can only be accessed by class A methods, so only x,w,y, and z

(i) Since inB is an object, it invokes an instantiation of method x.

(j) Methods w(), x(), y(), r(), and s() are available to class B

(k) Class B clients can access methods w(), and x().